REMARKS

Applicant is in receipt of the Office Action mailed May 29, 2008. Claims 1-35 remain pending in the application. Reconsideration of the present case is earnestly requested in light of the following remarks.

Telephone Interview Summary

On Tuesday, August 26, 2008, a telephone interview was conducted between Examiner Bullock, Jeffrey C. Hood (Reg. #35,198), and Mark S. Williams (Reg.# 50,658). Applicants noted that the cited art of OS/2 Client/Server Toolkit ("OS/2"), originally cited in a section 103 rejection, was now used to make a section 102 rejection (in response to a filed Appeal Brief). Applicant argued that OS/2 was not directed to computer system component level models and instrumentation, but rather to monitoring various software processes in a database transaction system, e.g., login, event logging, etc. The Examiner agreed that OS/2 did not seem germane to Applicant's claimed invention, and suggested that 5 other references be analyzed to determine whether any of them would require changes to the independent claims. Applicants reviewed these new references and determined that they did not disclose the subject matter of Applicant's independent claims. The Examiner subsequently suggested that a Response be submitted with respect to the previously cited art of OS/2, and that the Response also indicate that the 5 additional references were analyzed.

35 U.S.C. §102 Rejections

Claims 1-10, 12-17 and 19-30 stand rejected under 35 U.S.C. §102(b) as being anticipated by "OS/2 Client/Server Toolkit", Angelo R. Bobak, 1995 ("OS/2").

As the Examiner is certainly aware, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American*

Hoist & Derrick Co., 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). (Emphasis added)

The Office Action states that "OS/2 teaches a management system for generation of a management object model including a structured hierarchy of objects representing components of a computer system for performing management of the computer system", citing OS/2, page 562, Figure 19.1

Applicant submits that page 562 of the OS/2 reference merely lists various global variables, local variables, and definitions for a "coding exercise" for a "transaction monitor". The OS/2 reference at page 561 describes the "transaction monitor" as "allow[ing] you to interface with either the memory version of the database or any of the databases that our packages support." (OS/2 page 561). Thus the "transaction monitor" being constructed is merely a database access tool that can accept and execute database transactions or database requests. Figure 19.1 does not appear to be described in the OS/2 reference. However, Figure 19.1 appears to merely illustrate software classes used in creating this database access program, or possibly the "transaction monitor" portion of the program that "accepts [database] transaction requests from a client, executes them, and returns the results to the client process." (OS/2 page 561).

Applicant can find no teaching or suggestion regarding a "management object model" or "a structured hierarchy of objects representing components of a computer system" or a "management object model . . . for management of the computer system". For example, nowhere can Applicant find objects that represent components (e.g., physical resources or components in the system). The software classes shown in Figure 19.1 correspond to various software tasks, such as "event logger", "login monitor", "time keeper", "admin panel", and "kernel" that are used in the database program being created.

With respect to the claim element "component modules operable to define mappings from instrumentation of the components to objects representing those components", the Office Action refers to OS/2 page 610 and Figure 22.1 (Chapter 22).

This chapter of the OS/2 reference is also involved with creating software for the database access program begun in Chapter 19. Chapter 22 describes creation of "system configuration logic" and "keyboard logic". The "keyboard logic" is simply software which monitors the keyboard and allows the administrator to enter keystrokes that are interpreted and executed. The Office Action appears to rely on Figure 22.1. Again, Figure 22.1 shows a more detailed software class diagram of the database access program ("transaction monitor") that is being constructed in this exercise. In other words, Figure 22.1 merely shows the software classes used in constructing a database access program ("transaction monitor"). As merely one example, Applicant cannot find any "objects representing those components" or any "component modules operable to define mappings from instrumentation of the components to objects representing those components". In short, this software class diagram for a database program is simply not relevant to the claimed subject matter.

With respect to the claim element "configuration modules operable to configure associations between the component modules for the generation of the management object model", the Office Action refers again to OS/2 page 562 Figure 19.1 and pages 610-619. The Office Action then states "OS/2 teaches the defining of system objects and being able to generate a management object model." Applicant respectfully submits that this is incorrect. The cited reference in general teaches a student how to create a database transaction program in OS/2. Pages 610-619 appear to teach creation of a "configuration parameter tool" used to create "monitor configuration parameters", i.e., parameters to configure the database transaction program. Applicant submits that the cited reference is simply irrelevant to the present claims.

Applicant submits that OS/2 fails to teach or suggest all the elements of claim 1. Specifically, OS/2 fails to teach, "a processor; and a memory coupled to the processor, wherein the memory comprises program instructions configured to implement: component modules operable to define mappings from instrumentation of the components to objects representing those components, and configuration modules

operable to configure associations between the component modules for the generation of the management object model" as recited by claim 1.

In accordance, claim 1 is believed to patentably distinguish over OS/2. Likewise, independent claims 20, 21, and 35 recite features similar to those highlighted above with regard to independent claim 1, and are therefore believed to patentably distinguish over OS/2 for at least the reasons given in the above paragraphs discussing claim 1.

Claims 2-19 are dependent upon claim 1 and are therefore believed to patentably distinguish over the cited reference for at least the same reasons. Similarly, claims 22-34 are dependent upon claim 21 and are therefore believed to patentably distinguish over the cited reference for at least the same reasons. Moreover, Applicant submits that numerous of these claims recite further novel limitations over the cited art of OS/2.

For example:

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein said component modules are operable to define mappings at respective different levels of abstraction**, as recited by claim 2.

As explained above with reference to claim 1, cited OS/2, p.562 lists various global variables, local variables, and definitions for a "transaction monitor" that allows a user to interface with a database, i.e., a database access tool. Nowhere does the cited text, nor OS/2 in general, disclose or describe component modules that define mappings from instrumentation of the components to objects representing those components at respective different levels of abstraction.

Cited Figure 19.1, which does not appear to be described in the OS/2 reference, illustrates software classes used in creating the "transaction monitor" database access program. Appellant can find no teaching or suggestion regarding component modules that define mappings from instrumentation of components to objects representing those components at all, much less where the component modules define the mappings *at respective different levels of abstraction*.

Examples of Appellant's component modules (with different levels of abstraction) are described in the Specification at p.2:15-19, where, for example, a component module defines a mapping for a single component property at a first level of abstraction, another component module defines a mapping for a set of component properties forming an object at a second level of abstraction, and a further component module defines a mapping for an assembly of associated objects at a third level of abstraction. As noted above, OS/2 fails to disclose objects that represent computer system components (e.g., physical resources or components in the system) at all, nor component modules that define mappings from instrumentation of the components to objects representing those components, nor, more specifically, that define the mappings *at respective different levels of abstraction*. Appellant notes that the software classes shown in Figure 19.1 correspond to various software tasks, such as "event logger", "login monitor", "time keeper", "admin panel", and "kernel" that are used in the database access program being created, and do not refer to computer system components, nor objects representing such components.

Thus, the cited art fails to teach or suggest this feature of claim 2, and so claim 2 is patentably distinct over the cited reference.

Similar arguments apply to claim 22, and so for at least the above reasons, claim 22 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said component module is operable to define a mapping for a single component property at a first level of abstraction**, as recited by claim 3.

As discussed above, cited OS/2, p.562-563 (which includes Figure 19.1) lists various global variables, local variables, and definitions for a "transaction monitor" that allows a user to interface with, i.e., access, a database, and further illustrates software classes used in creating the "transaction monitor" database access program (Figure 19.1). Nowhere do the cited portions of OS/2, nor OS/2 in general, disclose or describe component modules that define mappings from instrumentation of the components to objects representing those components at respective different levels of abstraction, nor, more specifically, a component module that defines a mapping for a single component

property at a first level of abstraction. Rather, the cited OS/2 portions disclose program global structures, variables, definitions, header files, and (Figure 19.1) software classes used to build the "transaction monitor" database access program, specifically, software classes corresponding to various software tasks, such as "event logger", "login monitor", "time keeper", "admin panel", and "kernel" that are used in the database access program. These classes, structures, variables, and definitions do not correspond to physical computer system components, nor do they define a mapping for a single computer system component property at a first level of abstraction.

Thus, the cited art fails to teach or suggest this feature of claim 3, and so claim 3 is patentably distinct over the cited reference.

Similar arguments apply to claim 23, and so for at least the above reasons, claim 23 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said component module is operable to define a mapping for a set of component properties forming an object at a second level of abstraction**, as recited by claim 4.

As discussed above, cited OS/2, p.562 (which includes Figure 19.1) lists various global variables, local variables, and definitions for a "transaction monitor" that allows a user to interface with, i.e., access, a database, and further illustrates software classes used in creating the "transaction monitor" database access program (Figure 19.1). Nowhere do the cited portions of OS/2, nor OS/2 in general, disclose or describe component modules that define mappings from instrumentation of the components to objects representing those components at respective different levels of abstraction, nor, more specifically, a component module that defines a mapping for a set of component properties forming an object at a second level of abstraction. Rather, the cited OS/2 portion discloses program global structures, variables, definitions, header files, and (Figure 19.1) software classes used to build the database access program, specifically, software classes corresponding to various software tasks, such as "event logger", "login monitor", "time keeper", "admin panel", and "kernel" that are used in the database access program. These classes, structures, variables, and definitions do not correspond to

physical computer system components, nor do they define a mapping for a set of component properties forming an object at a second level of abstraction.

Thus, the cited art fails to teach or suggest this feature of claim 4, and so claim 4 is patentably distinct over the cited reference.

Similar arguments apply to claim 24, and so for at least the above reasons, claim 24 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said component module is operable to define a mapping for an assembly of associated objects at a third level of abstraction**, as recited by claim 5.

As discussed above, cited OS/2, p.562 (which includes Figure 19.1) lists various global variables, local variables, and definitions for a "transaction monitor" that allows a user to interface with, i.e., access, a database, and further illustrates software classes used in creating the "transaction monitor" database access program (Figure 19.1). Nowhere do the cited portions of OS/2, nor OS/2 in general, disclose or describe component modules that define mappings from instrumentation of the components to objects representing those components at respective different levels of abstraction, nor, more specifically, a component module that defines a mapping for an assembly of associated objects at a third level of abstraction. Rather, the cited OS/2 portion discloses program global structures, variables, definitions, header files, and (Figure 19.1) software classes used to build the "transaction monitor" database access program, specifically, software classes corresponding to various software tasks, such as "event logger", "login monitor", "time keeper", "admin panel", and "kernel" that are used in the database access program. These classes, structures, variables, and definitions do not correspond to physical computer system components, nor do they define a mapping for an assembly of associated objects at a third level of abstraction.

Thus, the cited art fails to teach or suggest this feature of claim 5, and so claim 5 is patentably distinct over the cited reference.

Similar arguments apply to claim 25, and so for at least the above reasons, claim 25 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said component module for a component defines a behavior of the object representing the component**, as recited by claim 6.

The Office Action asserts that this feature is inherent in the definition of "object", since "objects are made of attributes and the methods to perform operations on those attributes. methods [*sic*] are the behavior."

Appellant respectfully submits that, as explained above, Appellant's claimed component modules define mappings from instrumentation of the components to objects representing those components. Thus, an object represents a component (of a computer system), and the corresponding component module defines mapping from instrumentation of the component to the object.

This particular relationship is neither taught nor suggested by the (well-known) objects utilized in object oriented systems, nor by the methods of objects in an objected oriented system. Appellant respectfully submits that since the cited art fails to teach or suggest objects that represent components, the art also cannot teach or suggest a component module that defines mapping from instrumentation of the component to such an object. Nor, more specifically, does the cited art disclose a component module that defines a behavior of the object representing the component.

Thus, the cited art fails to teach or suggest this feature of claim 6, and so claim 6 is patentably distinct over the cited reference.

Similar arguments apply to claim 26, and so for at least the above reasons, claim 26 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said configuration module is operable to configure a said component module dynamically at run time for a said component that is subject to dynamic changes in status and is further operable to monitor said component for a change in status**, as recited by claim 7.

The Office Action contends that the above-quoted feature of claim 7 is disclosed on pages 609 and 611-615, citing the configuration parameter tool (bottom of page 612). Appellant respectfully disagrees at least for the reasons cited above, since OS/2 does not

disclose component modules that represent computer system components, nor configuration modules for configuring such component modules. Moreover, as noted above, OS/2 teaches "The configuration logic will come in the form of two utilities: One allows you to create the file containing the monitor configuration parameters; the other utility allows you to look at the parameters by loading the file." (OS/2, page 609). Again, these are utilities being created for a database transaction or access program. This cited portion of OS/2 has nothing to do with a system component or resource (physical component or resource) that is subject to dynamic changes in status, and dynamic configuration of a component module for such a physical component, as recited in the claim.

Accordingly, claim 7 is believed to patentably distinguish over the cited art.

Similar arguments apply to claim 27, and so for at least the above reasons, claim 27 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said configuration module is operable to configure a said component module statically at run time for a said component having static properties for a given invocation of the computer system**, as recited by claim 8.

The Office Action contends that the above-quoted feature of claim 8 is disclosed on pages 609 and 592, citing fixed properties such as the size of a message as defined to be 256 characters. Appellant respectfully disagrees at least for the reasons cited above, since OS/2 does not disclose component modules that represent computer system components, nor configuration modules for configuring such component modules. For example, a message is not a computer system component.

Moreover, per cited p.609, OS/2 teaches "The configuration logic will come in the form of two utilities: One allows you to create the file containing the monitor configuration parameters; the other utility allows you to look at the parameters by loading the file." Again, these are utilities being created for a database transaction or access program. This cited portion of OS/2 has nothing to do with a computer system component or resource (physical component or resource) that has static properties for a

given invocation of the computer system, and static configuration of a component module for such a physical component at runtime, as recited in the claim.

Accordingly, claim 8 is believed to patentably distinguish over the cited art.

Similar arguments apply to claim 28, and so for at least the above reasons, claim 28 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach program instructions implementing **wherein a said configuration module is operable to configure a said component module fixedly at run time for a said component having fixed properties for any invocation of the computer system**, as recited by claim 9.

The Office Action refers to the Examiner's arguments made with respect to claim 8, apparently asserting that pages 609 and 592 also disclose this feature of claim 9, citing fixed properties such as the size of a message as defined to be 256 characters. Appellant respectfully disagrees at least for the reasons cited above, since OS/2 does not disclose component modules that represent computer system components, nor configuration modules for configuring such component modules. As noted above, a message is not a computer system component.

Moreover, per cited p.609, OS/2 teaches "The configuration logic will come in the form of two utilities: One allows you to create the file containing the monitor configuration parameters; the other utility allows you to look at the parameters by loading the file." Again, these are utilities being created for a database transaction or access program. This cited portion of OS/2 has nothing to do with a computer system component or resource (physical component or resource) that has fixed properties for a given invocation of the computer system, nor fixedly configuring a component module for such a physical component at runtime, as recited in the claim.

Accordingly, claim 9 is believed to patentably distinguish over the cited art.

Similar arguments apply to claim 29, and so for at least the above reasons, claim 29 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **a library of component modules**, as recited by claim 10.

As discussed above, cited OS/2, p.562 (which includes Figure 19.1) lists various global variables, local variables, and definitions for a "transaction monitor" that allows a user to interface with, i.e., access, a database, and further illustrates software classes used in creating the "transaction monitor" database access program (Figure 19.1).

Appellant respectfully submits that no mention of a component library is made, nor is a component library shown, on p.562. However, while OS/2 may have a library of software modules, this in no way teaches or suggests a library of component modules, where the component modules define mappings from instrumentation of computer system components to objects representing those components. Software modules are not computer system components, i.e., physical components or resources. As noted earlier, the classes, structures, variables, and definitions of p.562, and the software classes used in creating this database access program shown in Figure 19.1 do not correspond to physical computer system components, nor do they define mappings from instrumentation of computer system components to objects representing those components.

Thus, the cited art fails to teach or suggest this feature of claim 10, and so claim 10 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein a said component module for a component identifies an instrumentation module defining a source of instrumentation for the component**, as recited by claim 12.

Cited OS/2 p.603 is directed to transaction objects and structures used to process transactions, specifically, structures used by transaction agents that execute database requests against IBM DB/2 databases, and has nothing to do with "instrumentation of components in the computer system". For example, the cited transRecords ("transaction records") is a union of structures that can be submitted to the monitor for transaction requests, e.g., carrierPacket, chargPacket, customerPacket, invoicePacket, packagePacket, and routePacket. None of the objects and structures is described as being a component module for a computer system component (physical resource or component of the computer system), nor an instrumentation module. More specifically, the cited

transRecord is nowhere described as a component module for a component that identifies an instrumentation module defining a source of instrumentation for the component. In fact, nowhere are computer system components, or component modules, or instrumentation (or instrumentation modules) mentioned at all. Thus, the cited art does not, and cannot, disclose a component module that identifies an instrumentation module defining a source of instrumentation for the component.

Thus, the cited art fails to teach or suggest this feature of claim 12, and so claim 12 is patentably distinct over the cited reference.

Similar arguments apply to claim 30, and so for at least the above reasons, claim 30 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein the instrumentation module exports an object-based representation of the instrumentation data via an instrumentation interface**, as recited by claim 13.

Cited OS/2 p.627 summarizes the coding exercise "transaction monitor" description and toolkit used to build it, and mentions a database layer that simulates a database in memory for use by readers who do not have access to LAN-based platforms such as Gupta or Microsoft/Sybase. Figure 22.2 is a portion of a sample screenshot of the "Transaction Monitor in Action", showing various "Expert System" steps, presumably performed by the transaction monitor, but not explained in the text.

Nowhere do the citations mention or even hint at an instrumentation module, nor, more specifically, an instrumentation module that exports an object-based representation of instrumentation data via an instrumentation interface. In fact, the citations make no mention of instrumentation, an instrumentation module, or instrumentation data at all.

Thus, the cited art fails to teach or suggest this feature of claim 13, and so claim 13 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein the instrumentation module comprises a general part and a specific part, the general part being operable to communicate with the specific part via a private interface to obtain instrumentation data, and the specific part being**

configured to interface with instrumentation for the component to obtain said instrumentation data, as recited by claim 14.

Cited OS/2 p.618-619 discloses various keyboard agent functions for handling user input key commands. For example, pop-up panels can be created for the database program to display an event log. This has nothing to do with instrumentation or user interfaces for hardware components, and certainly does not teach the specifics of claim 14. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data. Such instrumentation of a computer system component is not discussed in the cited reference.

Accordingly, claim 14 is believed to patentably distinguish over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein the general part and the specific part are local to each other**, as recited by claim 15.

Cited OS/2 p.612 discloses portions of a configuration parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. P.613 continues the description, where the user is prompted for parameter values, the structure is populated with values received from the user, and a corresponding record is written to the newly created configuration file. Appellant notes that the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local with respect to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and

more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data, where the general part and the specific part are local or remote to each other.

Thus, the cited art fails to teach or suggest this feature of claim 15, and so claim 15 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein the specific part is remote from the general part, the general part being operable to communicate with the remote part via a remote access mechanism**, as recited by claim 16.

In the rejection of claim 16, the Examiner refers to the arguments made with respect to claim 15. However, as noted above, cited OS/2 p.612 discloses portions of a configuration parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. As also noted above, the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local with respect to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are remote or local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system, and fails to disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data, where the specific part is remote from the general part, and where the general part communicates with the remote part via a remote access mechanism.

Thus, the cited art fails to teach or suggest this feature of claim 16, and so claim 16 is patentably distinct over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **comprising a library of instrumentation modules**, as recited by claim 17.

Cited OS/2 p.610 and Figure 22.1 disclose a class diagram for the code exercise "transaction monitor" discussed above, including software classes such as the transaction monitor itself, request monitor, login monitor, event logger, time keeper, admin panel, and so forth. The citations also describe session configuration parameters, such as the name of the login pipe with its input and output size, location of the monitor, e.g., REMOTE or LOCAL, with respect to the client process.

Nowhere does the cited text (nor OS/2 in general) disclose instrumentation modules, nor a library of such modules.

Accordingly, claim 17 is believed to patentably distinguish over the cited reference.

Appellant respectfully submits that OS/2 fails to teach a management system that includes **wherein the management system forms a management agent for remote management of a computer system**, as recited by claim 19.

Cited OS/2 p.603-604 discloses a modified transaction object referred to as transaction records, which is a union of various structures needed to support transaction agents that execute requests against the database. Applicant respectfully submits that the transaction monitor cited by the Examiner is for performing database operations, e.g., database requests, and is not germane to remote management of a computer system.

Accordingly, claim 19 is believed to patentably distinguish over the cited reference.

Applicant also asserts that numerous other ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 102 rejection of claims 1-10, 12-17 and 19-30 is earnestly requested.

35 U.S.C. §103 Rejections

Claims 11, 18, 31, 32, 33, and 34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over "OS/2 Client/Server Toolkit", Angelo R. Bobak, 1995 (hereinafter "OS/2") in view of Lorenz et al. (U.S. Patent No. 6,405,366).

Applicant notes that since independent claims 1, 20, 21, and 35, were shown above to be patentably distinct and non-obvious, and thus allowable, their respective dependent claims are similarly patentably distinct and non-obvious, and thus allowable, for at least the reasons provided above.

Moreover, Applicant submits that numerous of these claims include further distinctions over the cited art. For example:

Appellant respectfully submits that OS/2 and Lorenz, taken singly or in combination, fail to teach a management system that includes **wherein a said component module comprises a plug-in module**, as recited by claim 11.

Cited OS/2 p.603-604 discloses a modified transaction object referred to as transaction records, which is a union of various data structures needed to support transaction agents that execute requests against the database. Applicant respectfully submits that the transaction monitor cited by the Examiner is for performing database operations, e.g., database transactions/requests, and is not germane to remote management of a computer system. Moreover, the cited plugins of Lorenz are specifically for accessing formatted data of respective data types, e.g., binary, symbolic, etc., and have nothing whatsoever to do with a component module that defines mappings from instrumentation of computer system components to objects representing those componentst. Thus, even in combination with the plugins of Lorenz, the combination of OS/2 and Lorenz fail to teach or suggest the features of claim 11.

Accordingly, claim 11 is believed to patentably distinguish over the cited references.

Appellant respectfully submits that OS/2 and Lorenz, taken singly or in combination, fail to teach a management system that includes **wherein a said instrumentation module comprises a plug-in module**, as recited by claim 18.

Cited OS/2 p.603-604 discloses a modified transaction object referred to as transaction records, which is a union of various data structures needed to support transaction agents that execute requests against the database. Applicant respectfully submits that the transaction monitor cited by the Examiner is for performing database operations, e.g., database requests, and is not germane to remote management of a computer system. Moreover, the cited plugins of Lorenz are specifically for accessing formatted data of respective data types, e.g., binary, symbolic, etc., and have nothing whatsoever to do with an instrumentation module that identifies an instrumentation source for a computer system component. Thus, even in combination with the plugins of Lorenz, the combination of OS/2 and Lorenz fail to teach or suggest the features of claim 18.

Accordingly, claim 18 is believed to patentably distinguish over the cited references.

Appellant respectfully submits that OS/2 and Lorenz, taken singly or in combination, fail to teach **the instrumentation module exporting an object-based representation of the instrumentation data via an instrumentation interface**, as recited by claim 31.

Cited OS/2 p.627 summarizes the coding exercise "transaction monitor" description and toolkit used to build it, and mentions a database layer that simulates a database in memory for use by readers who do not have access to LAN-based platforms such as Gupta or Microsoft/Sybase. Figure 22.2 is a portion of a sample screenshot of the "Transaction Monitor in Action", showing various "Expert System" steps, presumably performed or reported by the transaction monitor, but not explained in the text.

Nowhere do the citations mention or even hint at an instrumentation module, nor, more specifically, and instrumentation module exporting an object-based representation of instrumentation data (of a computer system component) via an instrumentation

interface. In fact, the citations make no mention of instrumentation, an instrumentation module, or instrumentation data at all.

As noted above, the cited plugins of Lorenz are specifically for accessing formatted data of respective data types, e.g., binary, symbolic, etc., and have nothing whatsoever to do with an instrumentation module that identifies an instrumentation source for a computer system component. Thus, even in combination with the plugins of Lorenz, the combination of OS/2 and Lorenz fail to teach or suggest the features of claim 31.

Thus, the cited art fails to teach or suggest this feature of claim 31, and so claim 31 is patentably distinct over the cited references.

Appellant respectfully submits that OS/2 and Lorenz, either singly or in combination, fail to teach **a general part of the instrumentation module communicating with a specific part of the instrumentation module via a private interface to obtain instrumentation data, and the specific part interfacing with instrumentation for the component to obtain said instrumentation data**, as recited by claim 32.

Cited OS/2 p.618-619 discloses various keyboard agent functions for handling user input key commands. For example, pop-up panels can be created for the database program to display an event log. This has nothing to do with instrumentation or user interfaces for computer system components (hardware), and does not teach or suggest the specifics of claim 32. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data. Such instrumentation of a computer system component is not discussed in the cited references.

The Examiner relies on the arguments made with respect to claim 14. As discussed above with respect to that claim, cited OS/2 p.618-619 discloses various keyboard agent functions for handling user input key commands. For example, pop-up

panels can be created for the database program to display an event log. This has nothing to do with instrumentation or user interfaces for hardware components, and certainly does not teach the specifics of claim 14. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data. Such instrumentation of a computer system component is not discussed in the cited reference.

Accordingly, claim 32 is believed to patentably distinguish over the cited references.

Appellant respectfully submits that OS/2 and Lorenz, either singly or in combination, fail to teach a management system that includes **wherein the general part and the specific part are local to each other**, as recited by claim 33.

Cited OS/2 p.612 discloses portions of a configuration parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. P.613 continues the description, where the user is prompted for parameter values, the structure is populated with values received from the user, and a corresponding record is written to the newly created configuration file. Appellant notes that the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the

component to obtain the instrumentation data, *where the general part and the specific part are local or remote to each other.*

The Examiner relies on the arguments made with respect to claim 15. As discussed above with respect to that claim, cited OS/2 p.612 discloses portions of a configuration parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. P.613 continues the description, where the user is prompted for parameter values, the structure is populated with values received from the user, and a corresponding record is written to the newly created configuration file. Appellant notes that the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local with respect to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system. Nor does this text (nor OS/2 in general) disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data, where the general part and the specific part are local or remote to each other.

Thus, the cited art fails to teach or suggest this feature of claim 33, and so claim 33 is patentably distinct over the cited references.

Appellant respectfully submits that OS/2 and Lorenz, either singly or in combination, fail to teach a management system that includes **wherein the specific part is remote from the general part, the general part being operable to communicate with the remote part via a remote access mechanism**, as recited by claim 34.

In the rejection of claim 34, the Examiner refers to the arguments made with respect to claim 16, which refer to the arguments made with respect to claim 15. However, as noted above, cited OS/2 p.612 discloses portions of a configuration

parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. As also noted above, the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local with respect to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are remote or local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system, and fails to disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data, *where the specific part is remote from the general part, and where the general part communicates with the remote part via a remote access mechanism.*

The Examiner relies on the arguments made with respect to claim 16, which rely on the arguments made cited with respect to claim 15. As noted above, cited OS/2 p.612 discloses portions of a configuration parameter tool that facilitates instantiation of a configuration file/structure that includes initialization of parameters such as pipe name, pipe sizes, and transaction pipe sizes, as well as locality information regarding the server. As also noted above, the locality information (REMOTE or LOCAL) refers to whether the server location is remote or local with respect to the client process, *not* whether two parts of an instrumentation module (general and specific parts) are remote or local to each other.

This text does not describe and is not germane to instrumentation for a hardware component of a computer system, and fails to disclose an instrumentation module or instrumentation data from a computer system component, and more particularly, an instrumentation module that includes a general part and a specific part, where the general part communicates with the specific part via a private interface to obtain instrumentation data, and the specific part interfaces with instrumentation for the component to obtain the instrumentation data, where the specific part is remote from the general part, and where the general part communicates with the remote part via a remote access mechanism.

Thus, the cited art fails to teach or suggest this feature of claim 34, and so claim 34 is patentably distinct over the cited references.

Removal of the section 103 rejection of claims 11, 18, 31, 32, 33, and 34 is earnestly requested.

Additional References

The Examiner suggested that the following references also be analyzed with respect to Applicant's independent claims:

Zager et al. (U.S. Patent No. 6,393,386, "Zager")
Hayball et al. (U.S. Patent No. 6,349,332, "Hayball")
Battacharjee et al (U.S. Patent No. 7,284,163, "Battacharjee")
Muhlestein et al. (U.S. Patent No. 6,996,809, "Muhlestein '809")
Muhlestein et al. (U.S. Patent Pub. 200220108102, "Muhlestein '102")

Applicant has reviewed these references and believes that the features and limitations of the independent claims are not disclosed therein, for at least the following reasons.

Zager discloses a model that represents resources of a complex system, e.g., a distributed computing ensemble, and that also represents service-relationships among the resources. Managed resources are monitored by agents that can record performance data for the resource, report a change of state to or from an anomalous state (a fault), or create a new managed object to represent a previously unknown resource.

The only hierarchies mentioned are a data gathering infrastructure, which "is conceptually distinct from and independent of the model", and an object inheritance hierarchy, discussed briefly as an undesirable approach. Zager does not explicitly disclose a structured hierarchy of objects representing components of a computer system.

Zager discloses instrumentation of resources/components, but only with respect to supporting automatic discovery of the components. Zager discloses agents that operate to monitor changes in system objects, but makes no mention of component modules that define mappings from instrumentation of the components to (software) objects representing those components.

Zager discusses modeling system components and their service relationships, but doesn't appear to disclose configuration modules that configure associations between such component modules for the generation of a management object model.

Thus, Zager does not appear to teach or suggest these claimed features.

Hayball is directed to a managed object based MIB (management information base) that separates the representation of physical/logical resources from representation of the functionality of these resources, but doesn't appear to be germane to the claimed subject matter. For example, no mention is made of instrumentation of computer system components, component modules for mapping instrumentation of such components to objects representing those components, or configuring associations between such component modules for the generation of the management object model.

Thus, Hayball does not appear to teach or suggest these claimed features.

Bhattacharjee is directed to an instrumentation data provider module for use within a diagnostics application program, specifically, to provide an interface between legacy diagnostic modules and a standard instrumentation platform by converting data to and from formats to facilitate communication between the two.

The instrumentation platform includes "programs for modeling components within a computer system and for receiving configuration, status, and operational information from the components." However, no mention is made of a management object model that includes a structured hierarchy of objects representing components of the computer system.

Bhattacharjee nowhere discloses or even hints at any type of mapping, and more specifically fails to describe component modules that define mappings from instrumentation of the components to objects representing those components. In other

words, Bhattacharjee does not appear to utilize such component models to implement or specify the relationship between instrumentation of components and component models. Nor does Bhattacharjee appear to disclose configuration modules for configuring associations between such modules for generation of a management object model.

Thus, Bhattacharjee does not appear to teach or suggest these claimed features.

Muhlestein '809 is directed to provision of instrumentation data within a managed code runtime environment to an external instrumentation data source by a "decoupled provider". The decoupled provider registers schema for managed code objects to be instrumented at run-time, and facilitates invocation of methods on instrumented data that reside within the managed code environment by an instrumentation data source executing outside the managed code environment, such as Microsoft™ Windows Management Instrumentation (WMI), a set of system services and programming interfaces that allow applications to expose instrumentation data in a consistent manner. While Muhlestein '809 discloses an interface for an external instrumentation data source communicating instrumentation data to and from objects in a managed code environment, nowhere does Muhlestein '809 disclose modules that define mapping from instrumentation components to objects representing components of a computer system, nor any sort of mapping at all. Nor does Muhlestein '809 appear to describe configuration modules for configuring associations between the component modules for the generation of a management object model.

Thus, Muhlestein '809 does not appear to teach or suggest these claimed features.

Muhlestein '102 is directed to an API for accessing instrumentation data outside a managed code runtime environment in a manner consistent with the managed code runtime platform. Muhlestein '102 does not appear to disclose any sort of module or program that defines mapping from instrumentation components to objects representing components of a computer system, nor any sort of mapping at all. Nor does Muhlestein '102 appear to describe configuration modules for configuring associations between the component modules for the generation of a management object model. Rather, Muhlestein '102's instrumentation data are provided by an "instrumentation data source",

such as Microsoft™ Windows Management Instrumentation (WMI), which "is a set of system services and programming interfaces that allow applications to expose instrumentation data in a consistent way."

Thus, Muhlestein '102 does not appear to teach or suggest these claimed features.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5681-76400/JCH.

Also filed herewith are the following items:

☐ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-08-29 JCH/MSW